AGENDA

1. Introduction
2. Background
   1. Data Mining
   2. Web Scraping
3. Simple Web Scraping Example – Tattoo Shop
4. PasteHunter
   1. Tool Architecture
   2. Paste Site Info
   3. Takeaway Concepts
5. General Thoughts
6. Questions
whoami

- @cham423 (keybase, I don’t use Twitter)
  - Discord cham423#2790, github
- Offensive Security @ Optiv (4 years)
  - Adversary Sim, R&D, Consulting
- Pentesting since 2013
- Networking Degree
- Hobbies: hoarding data, electronic music, cars, mountain biking
Who This Talk is For

- **Red Teamers:** Collect operationally useful info
- **Blue Teamers:** Be informed about information exposure and leaks
- **Anyone else:** Learn data mining concepts and use them to make your life better

- **Helpful Background Knowledge:**
  - Python / General Coding Concepts
  - Linux
  - Elasticsearch
  - OSINT
What is Data Mining?

- Buzzword alert!
- I use this term to refer to "any form of large-scale data processing" including:
  - Collection, extraction, warehousing, analysis
- Collecting data is relatively easy
- Extracting value from data is harder
- Analyzing it is even harder
- Creating actionable data flows is the hardest of all
What about Web Scraping?

- A specific kind of data collection + extraction focused on web pages

- Techniques:
  - Retrieve page, parse HTML (Python Requests + BeautifulSoup)
  - Use scraping library (Python Scrapy)
  - Simulate real browser and user actions (Selenium, webbrowser)

- At scale, expect countermeasures

- Honor among bots

- Third parties (Bright Data, Apify, etc)
GETTING TATTOOS USING PYTHON

Kinda...
The Problem

- The best tattoo artists are booked for months or years
- They periodically open their books to fill gaps, then close them
- To get a booking, you must monitor their websites/social media profiles for updates
The Solution

- [https://github.com/cham423/simple-scraping-example](https://github.com/cham423/simple-scraping-example)
- <100 lines of code
- Uses requests, bs4 to scrape site and determine state of books (open or closed)
- Notifies user with Pushover
- Stores results in Elasticsearch
- Hacky
Request + Parsing

```python
response = requests.get(url, headers=headers)
# parse the downloaded homepage
soup = BeautifulSoup(response.text, "lxml")
# hash the content from the div class=paragraph objects.
# you will likely have to adjust this to a different HTML attribute, depending on the content you are trying to scrape
# this hash is useful to validate whether the content changed or not, since we are really only looking for specific strings
# in general you should build in fail-safes, because regex isn't always reliable, and web content changes a lot
contentHash = hashlib.sha256(str(soup.findAll('div', attrs={"class":"paragraph"})).encode('utf-8')).hexdigest()
# compare last hash to current hash, then send a notification if it changes
if (lastHash is not None) and (contentHash != lastHash):
    sendPushoverNotification("site content changed, check {}".format(url))
```

- Set request headers to real user agent (optional)
- Parse with BeautifulSoup “lxml” parser
- Hash the content, notify user if it changes
- Will have to be tailored to your target site
- More difficult than it sounds
- Be “greedy” with data collection
- Build in failsafes
Regex + Logic

```python
# search for strings in the content with regex.
# if the string is found at least once, we assume the books are closed
if len(soup.findAll(string=re.compile("books closed", re.IGNORECASE))) > 0:
    isOpen = False
# if the string isn't found at least once, let's check if the books are open
elif len(soup.findAll(string=re.compile("books open", re.IGNORECASE))) > 0:
    isOpen = True
# if neither string is found, either the website is down or the content of the page has
else:
    # send notification so we know that something went wrong
    sendPushoverNotification("Something is up, scraper exiting, check {}\".format(url))
    sys.exit("couldn't determine state, exiting")
```

- "I should learn regex" – everyone
- Don’t trust it completely
- Think through the logic (mine is questionable)
Structure + Index

```
class ShopState:
    def __init__(self, timestamp, isOpen, contentHash):
        self.timestamp = timestamp
        self.isOpen = isOpen
        self.contentHash = contentHash

    # now that we have data, let's do something with it
    currentState = ShopState(checkTime, isOpen, contentHash)
    # log the actual json info (optional)
    logger.info(json.dumps(currentState.__dict__))
    # index to elasticsearch
    try:
        es.index(index=indexName, body=json.dumps(currentState.__dict__))
    except Exception as e:
        logger.error("error indexing to elasticsearch: {}".format(e))
```

- Define your data clearly
- Classes make working with your data easier
- Direct conversion to JSON with `.__dict__`
  - Caveats w/ certain types
Results

- So far, no booking yet 😊
- Peace of mind
- Capturing data for future analysis:

```
<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>_id</td>
<td>uYCcZmkB84Bhxz31aKIDN</td>
</tr>
<tr>
<td>_index</td>
<td>dev_scraper</td>
</tr>
<tr>
<td>_score</td>
<td>-</td>
</tr>
<tr>
<td>_type</td>
<td>_doc</td>
</tr>
<tr>
<td>contentHash</td>
<td>feb1da0d2eab2b12e76aa1f69b52538c9e1b3b15d4f942fc46952193a88b9f</td>
</tr>
<tr>
<td>timestamp</td>
<td>May 9, 2021 @ 12:37:37</td>
</tr>
<tr>
<td>Site content</td>
<td>changed, check <a href="https://www.jaxtheboxtattoos.com">https://www.jaxtheboxtattoos.com</a></td>
</tr>
<tr>
<td>Site content</td>
<td>changed, check <a href="https://www.jaxtheboxtattoos.com">https://www.jaxtheboxtattoos.com</a></td>
</tr>
<tr>
<td>Site content</td>
<td>changed, check <a href="https://www.jaxtheboxtattoos.com">https://www.jaxtheboxtattoos.com</a></td>
</tr>
<tr>
<td>Site content</td>
<td>changed, check <a href="https://www.jaxtheboxtattoos.com">https://www.jaxtheboxtattoos.com</a></td>
</tr>
</tbody>
</table>
```
THAT’S COOL, BUT WHAT ABOUT THE WHOLE SECURITY THING
Introducing: Pastehunter

- https://github.com/kevthehermit/PasteHunter
- Built by @KevTheHermit (DFIR background)
- Python + Yara
- Dockerized
- Architecture:
  - Scrape data from inputs (pastebin.com, github gists, etc)
  - Scan data with Yara (built-in or custom rules)
  - Send data to outputs (elasticsearch, splunk, files, email, slack, etc)
  - Optionally, run binary content in a sandbox or do follow-up actions
My Experience

- Started using tool in 2017
- 20+ million total documents indexed
- 600GB+ storage used across all indexes
- Warning..
  - Volume of Data
  - Content
- Use Yara
Input and Outputs

Inputs:
- Pastebin pastes
- Github public Gists
- Github commit log
- ix.io
- Slexy.org (defunct)
- Stack Exchange (all sites)
- Anything you can code

Outputs:
- Elasticsearch / Splunk
- Email/Slack Alerts
- Flat files
  - JSON, CSV, syslog
- Webhooks
Yara

rule core_keywords
{
    meta:
        author = "@KevTheHermit"
        info = "Part of PasteHunter"
        reference = "https://github.com/kevthehermit/PasteHunter"

    strings:
        $tango_down = "TANGO DOWN" wide ascii nocase
        $antisec = "antisec" wide ascii nocase
        $hacked = "hacked by" wide ascii nocase
        $onion_url = "/.*\.onion/"
        $nmap_scan = "Nmap scan report for" wide ascii nocase
        $enabled_sec = "enable secret" wide ascii nocase
        $enable_pass = "enable password" wide ascii nocase

    condition:
        any of them

- [https://virustotal.github.io/yara/](https://virustotal.github.io/yara/)
- Designed for malware sample identification
- Easy to grasp, hard to master
- Most common hits:
  - `db_connection` - 2.81%
  - `email_filter` – 1.72%
  - `db_structure` – 1.16%
  - `sw_bucket` – 1.1%
- 80-90% of documents do not match rules (good filtering)
- Can have multiple matches per document
Custom Yara Rules

- Strongly recommended
- Store all pastes for a short time to build patterns
- Browse popular pastes
```
import pandas as pd
from sqlalchemy import create_engine

db_config = {
    'user': '???????', # имя пользователя
    'pwd': 'SdfS2:d-d30pp', # пароль
    'host': '???????', # хост
    'port': 5555, # порт подключения
    'db': 'data-analyst-zen-project-db' # название базы данных
}

connection_string = 'postgresql://{}:{}@{}:{}/{}'.format(db_config['user'],
                                                        db_config['pwd'],
                                                        db_config['host'],
                                                        db_config['port'],
                                                        db_config['db'])

engine = create_engine(connection_string)
```
email_filter

- Script/code dumps
- Email list leaks
- Leak “free previews”
sw_bucket

- Looks for S3 bucket URLs
- Often combined with aws_api
- False positive prone, but worth keeping
- Duckyscript
- Catch your next vendor with terrible opsec
- Extremely rare (~100 documents /year)
password_list

- Lists of compromised users
- Extremely rare (~100 documents/year)
b64_exe

- Base64 encoded binary file
- Sandboxing
- Again, catch your next vendor with bad opsec

pastesite.keyword: gist.github.com

5/27/2021
Pastebin Scraping API

First:

Pastebin @pastebin · Apr 15, 2020
The scraping API has been discontinued due to active abuse by third parties for commercial purposes, such activity is prohibited by our current T&C’s, please see Section C, P.4.

Then:

Pastebin @pastebin · Apr 27, 2020
The Scraping #API works as usual for PRO users who already bought it in the past, the only change made is to scrape pastes with correct syntax highlighting.

https://github.com/kevthehermit/Paste Hunter/issues/112
https://github.com/kevthehermit/Paste Hunter/issues/116

- Scraping API lives on currently
  - Less Data Available
  - Legacy account holders only
- Enterprise API?
Pastehunter with no Pastebin?

- GitHub
  - Public Gists
  - Public Commit History
- Other Paste Sites
- StackExchange

```
pastesite.keyword: ix.io
```

```
t raw_paste

48410::2::1::0::valensis::f924c2428b0dd0c5c7c11e167e6dd5379::2017-02-15::valensis@gmx.us::0::
7.70.11::0::1::0::0::0::1::90::80::8AS1L2h-cY-1(1)"*::0::0::0::0078ueil::2017-02-15::29791::0::
Topolm1::1286d38939f4b4e69c87672f796ed7e0::2016-12-07::topolm1@outlook.com::0::
1.179.80::0::0::1::0::0::0::0::0::0::0::0::0::0

t scrape_url

http://lx.io/2021T
```
Volume Statistics

- Yara Matches Only, 1 year timeframe
- Github: 450k documents
- Gists: 66k documents
- Slexy (defunct): 51k documents
- Ix.io: 40k documents
- Pastebin: 17k documents
Elasticsearch Tips

- Highly Recommend Making an Index Template
  - store: true on raw_paste field
  - 5 shards for weekly index
- String Fields
  - Text fields are analyzed during indexing
    - Broken down into terms for partial matching
  - Keyword fields are not broken into terms
    - Built for complete matches
  - New Wildcard Field (7.9+)
    - Perfect for pastes
- Don’t be afraid to reindex
  - Elasticsearch Dump (NPM)
The Future of Pastes

- Paste site usage has dropped
- Social media taking over
  - Twitter, Discord, Telegram
  - API crawling possible, but more complex
- Private Sites/Forums
  - Strong countermeasures against scraping
- Scraping now mainstream business model
  - Scraping as a service
    - Apify, many others
  - Data Aggregation platforms
    - Zoominfo, Intelligence Companies
- Legal Status
  - Legal, but lawsuits cost $$$ (C&D has power)
Final Notes

- Can’t I just search for this data?
  - Github: yes
  - Paste sites: kinda
  - Deletion/Expiration of Pastes

- Takeaways
  - Build good OSINT habits
  - Automation goes a long way
  - Use tools to your advantage
Stay Tuned...

- Framework tool release planned for late 2021/early 2022:
  - Elasticsearch + Spark + Jupyter Data Aggregation
    - Data Breaches
    - OSINT Data
    - Passwords
    - Scraped Data
THANK YOU

Corey Ham
Questions?

P.S. - Follow me on Github for upcoming data aggregation and OSINT projects